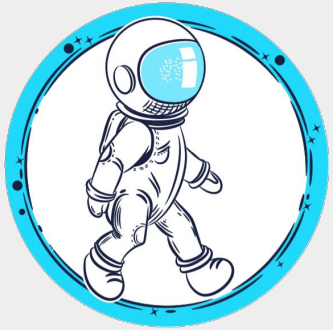
TRAINING RECAP

BY TOBIAS STÜBI

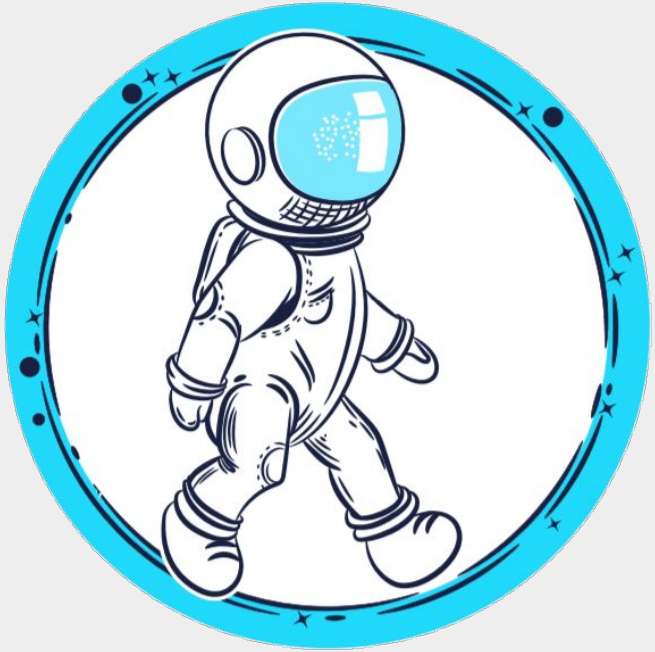


1. WALKING

2. RUNNING

3. FLYING

WALKING



MOB PROGRAMMING

NAVIGATOR

the decision maker

DRIVER

The Person typing on the keyboard

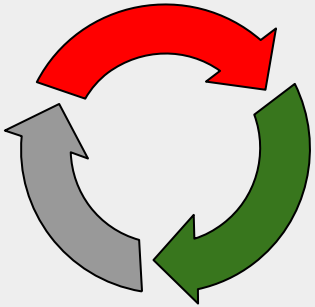
MOB

Everyone else in the room

TDD

RED

Write a failing test



GREEN

Write just enough code to pass the test

REFACTOR

Improve the code while passing the test

WHAT DO WE TEST?

WE TEST
BEHAVIOUR
NOT IMPLEMENTATION

ONE DEGREE OF FREEDOM
AT A TIME

TRANSFORMATION PRIORITY PREMISE

#	TRANSFORMATION	STARTING CODE	FINAL CODE
1	{} => nil		return nil
2	nil => constant	return nil	return “1”
3	constant => constant+	return “1”	return “1” + “2”
4	constant => scalar	return “1” + “2”	return argument
5	statement => statements	return argument	return arguments
6	unconditional => conditional	return arguments	if(condition)return arguments
7	scalar => array	dog	[dog, cat]
8	array => container	[dog, cat]	{dog = “DOG”, cat = “CAT”}
9	statement => recursion	a + b	a + recursion
10	conditional => loop	if(condition)	while(condition)
11	recursion => tail recursion	a + recursion	recursion
12	expression => function	today - birthday	CalculateAge()
13	variable => mutation	day	var day = 10; day = 11;
14	switch case		

OBJECT CALISTHENICS RULES

- ✓ Only one level of indentation per method
- ✓ Don't use the ELSE keyword
- ✓ Wrap all primitives and strings
- ✓ First class collections (wrap all collections)
- ✓ Only one dot per line dog.Body.Tail.Wag() => dog.ExpressHappiness()
- ✓ No abbreviations
- ✓ Keep all entities small *[10 files per package, 50 lines per class, 5 lines per method, 2 arguments per method]*
- ✓ No classes with more than two instance variables
- ✓ No public getters/setters/properties

RUNNING

REFACTORING WHEN?

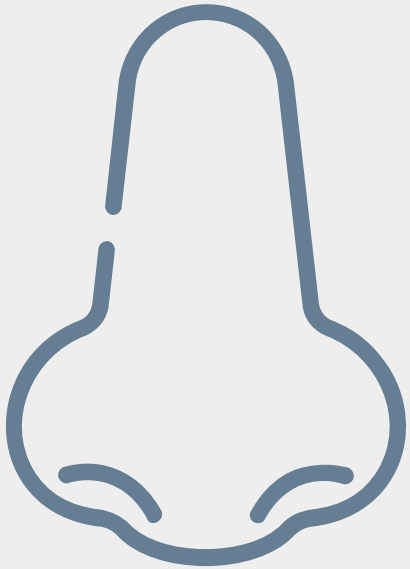
80 - 20 Rule

80% of the value in refactoring comes from improving readability

Rule of three

Breaking Object Calisthenics Rules

CODE SMELLS



https://sourcemaking.com/refactoring/smells

SOLID PRINCIPLES ++

Single Responsibility

Least Astonishment

Open/Closed

Liskov substitution

Dependency Inversion

Interface Segregation

COHESION AND COUPLING

Cohesion

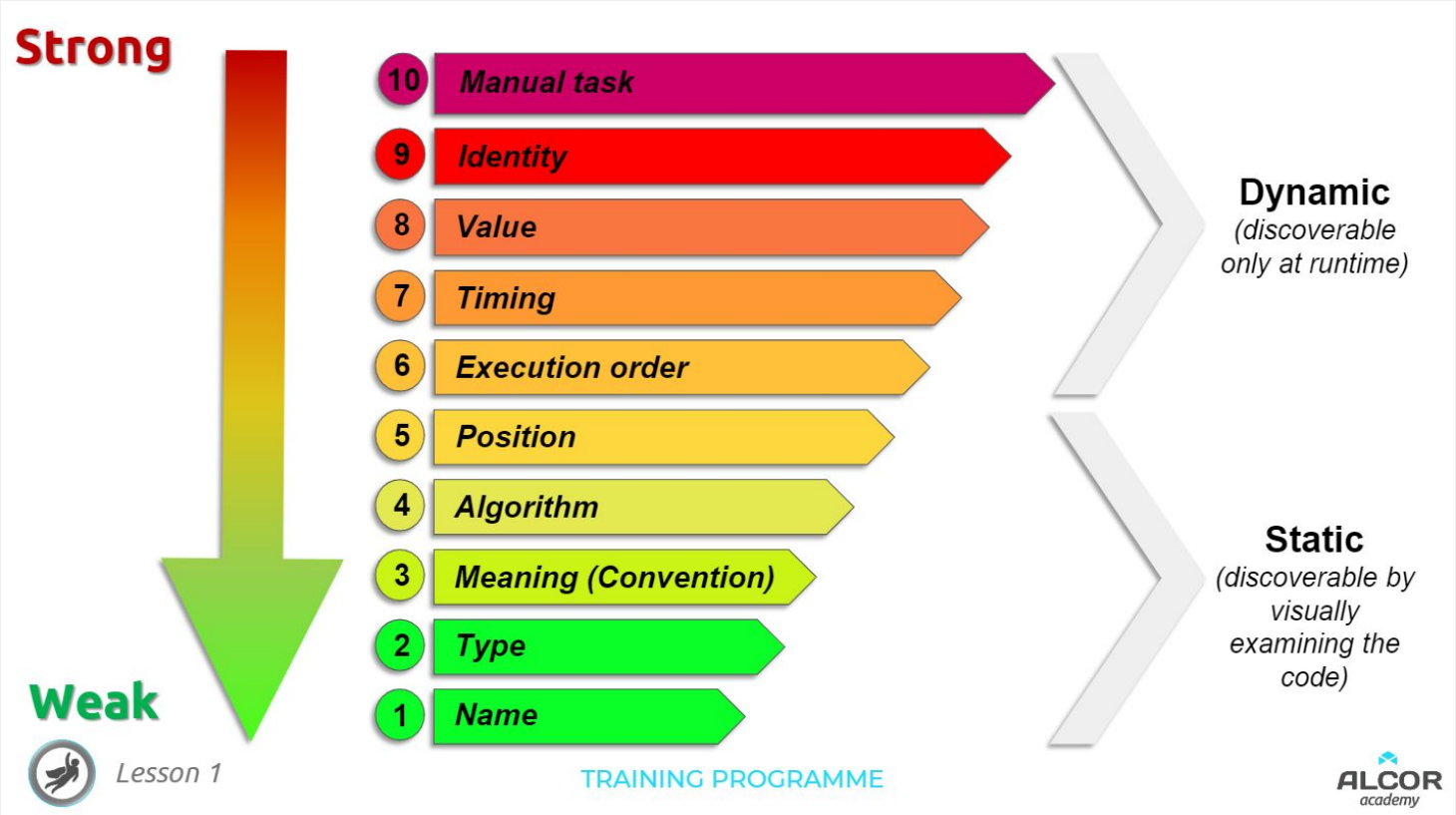
says how strongly related and coherent are the responsibilities within modules of an application

Coupling

is the degree of interdependence between modules of an application

FLYING

CONNASENCE


Strong
10 Manual task
9 Identity
8 Value
7 Timing
6 Execution order
5 Position
4 Algorithm
3 Meaning (Convention)
2 Type
1 Name
Weak
Dynamic
(discoverable only at runtime)
Static
(discoverable by visually examining the code)
Lesson 1
TRAINING PROGRAMME
ALCOR academy

TEST DOUBLES

DUMMY

needed to complete the parameters' list of a method, but never used. Not common in well-designed systems.

STUB

responds to calls with some pre-programmed output. They need to be specifically setup for every test. Fakes are handmade stubs.

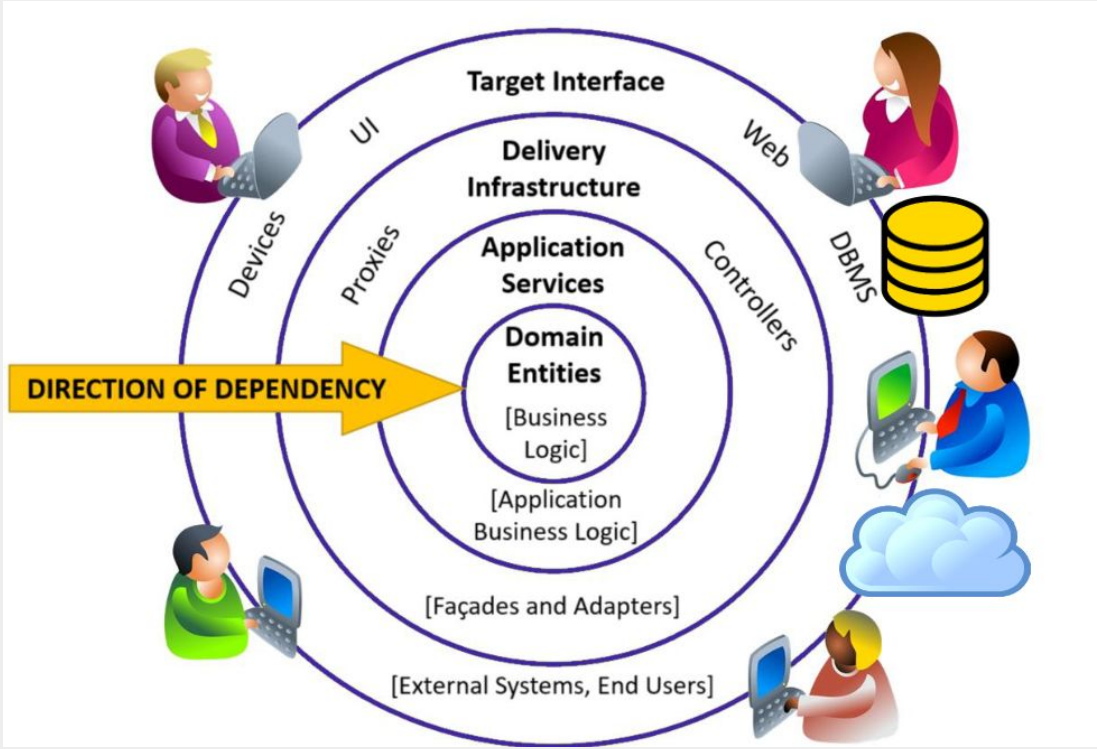
MOCK

set up with expectations of the calls they are expected to receive. Provide a way of verify that a behaviour has been triggered correctly. Spy is a handmade mock.

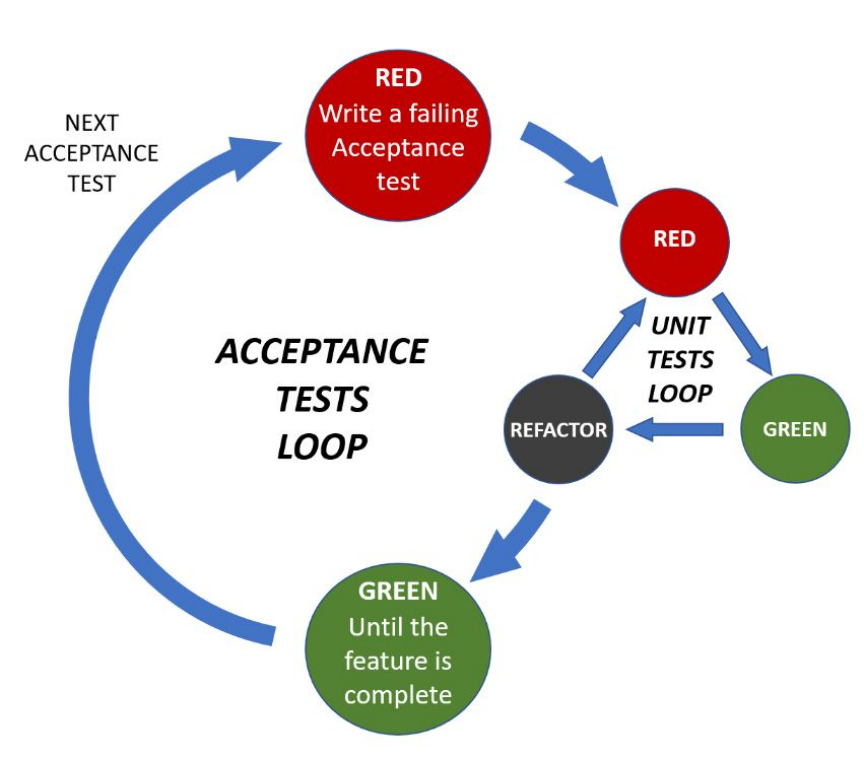
THE 4 RULES OF SIMPLE DESIGN

1. Passed the tests
2. Reveals intention
3. No duplication
4. Fewest elements

CLEAN / ONION / HEXAGONAL ARCHITECTURE



THE DOUBLE LOOP OF ATDD



THANKS!

Do you have any questions?

Contact:
stuebi@outlook.com